STAT 139: Final Project

Danny Kim, Christopher Lee, Karina Wang, Daniel Son

2022-12-14

EDA

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Load team data
team_data = list()
team_wins <- list()
drop = c("W", "L")
for (year in 1997:2022) {
  df1 = read.csv(paste("data/teams_data/batting", year, ".csv", sep=""))
  df2 = read.csv(paste("data/teams_data/pitching", year, ".csv", sep=""))
  df3 = read.csv(paste("data/teams_data/fielding", year, ".csv", sep=""))
  df_tot = merge(merge(df1, df2, by="Tm", suffixes=c(".bat", ".pitch")), df3, by="Tm", suffixes=c("", "

  df_tot = df_tot[
    !(df_tot$Tm %in% c("", "League Average")),
    !(names(df_tot) %in% drop)
  ]
  df_tot$Tm = factor(df_tot$Tm)
  team_data[[year]] = df_tot
  team_wins[[year]] = df_tot[, c("Tm", "W.L.")]
}

# Load player data
years <- 1997:2022
bps <- c("batting", "pitching", "fielding")
player_data <- list()
for (year in years) {
```

```
  player_data[[year]] <- list()
  for (bp in bps) {
    player_data[[year]][[bp]] <- read.csv(paste("data/player_data/", bp, year, ".csv", sep=""))
    quant_cols <- names(select_if(player_data[[year]][[bp]], is.numeric))
    for (col in quant_cols) {
      # impute data with mean
      df <- player_data[[year]][[bp]]
      player_data[[year]][[bp]][is.na(player_data[[year]][[bp]][,col]),col] <- mean(df[,col], na.rm=TRU
    }
  }
}

fa_data = list()
for (year in years) {
  fa_data[[year]] = read.csv(paste("data/fa_data/fa", year, ".csv", sep=""))
  fa_data[[year]]$WAR3[is.na(fa_data[[year]]$WAR3)] = 0
}
```

```
# Data Cleaning for the Team Data
team_wins <- list()
for (year in years) {
  team_wins[[year]] <- team_data[[year]][!(team_data[[year]]$Tm %in% c("", "League Average")), c("Tm",
}
```

```
# Clean player data
for (year in years) {
  for (bp in bps) {
    player_data[[year]][[bp]]$year <- year
    player_data[[year]][[bp]]$year_adj <- year - 1997
  }
}

for (year in years) {
  player_data[[year]][["pitching"]] = player_data[[year]][["pitching"]][!is.infinite(player_data[[year]
}

long_team_names <- team_data[[year]][!(team_data[[year]]$Tm %in% c("", "League Average")),]$Tm
short_team_names <- c("ARI", "ATL", "BAL", "BOS", "CHC", "CHW", "CIN", "CLE", "COL", "DET",
                      "HOU", "KCR", "LAA", "LAD", "MIA", "MIL", "MIN", "NYM", "NYY", "OAK",
                      "PHI", "PIT", "SDP", "SFG", "SEA", "STL", "TBR", "TEX", "TOR", "WSN")
agg_data <- list()
for (year in years) {
  agg_data[[year]] <- list()
  for (bp in bps) {
    quant_cols <- names(select_if(player_data[[year]][[bp]], is.numeric))
    agg_data[[year]][[bp]] <- player_data[[year]][[bp]][, c("Tm", quant_cols)] %>%
      group_by(Tm) %>%
      summarise(across(quant_cols, ~weighted.mean(., w = G)))
    agg_data[[year]][[bp]] <- agg_data[[year]][[bp]][!(agg_data[[year]][[bp]]$Tm == "TOT"),]
    agg_data[[year]][[bp]]$long_Tm <- factor(
      agg_data[[year]][[bp]]$Tm,
      levels=short_team_names,
      labels=long_team_names
```

```
    )
  }
}
```

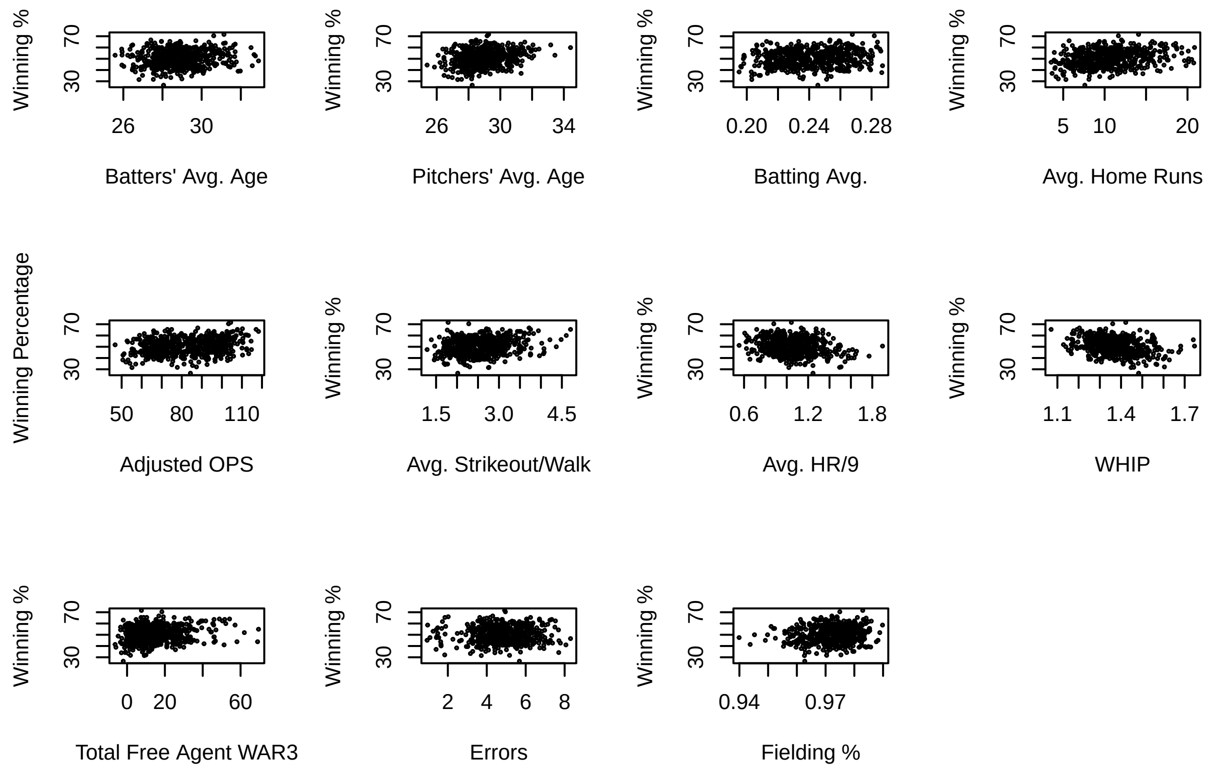
```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(quant_cols)
##
##   # Now:
##   data %>% select(all_of(quant_cols))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

```
player_combo <- list()
for (year in years) {
  player_combo[[year]] <- merge(merge(agg_data[[year]][[bps[1]]], agg_data[[year]][[bps[2]]], by="Tm", s
}

agg_fa_data <- list()
for (year in years) {
  agg_fa_data[[year]] = fa_data[[year]] %>% group_by(To.Team) %>% summarise(tot_fa_war3=sum(WAR3), num_
}

# add response variable to player data
player_with_wins <- list()
for (year in 1997:2021) {
  player_with_wins[[year]] <- merge(player_combo[[year]], team_wins[[year+1]], by.x="long_Tm.pitch", by
}

player_with_wins_fa <- list()
for (year in 1997:2021) {
  player_with_wins_fa[[year]] <- merge(player_with_wins[[year]], agg_fa_data[[year]], by.x="long_Tm.pit
}


player_with_wins_combined = bind_rows(player_with_wins_fa, )
player_with_wins_combined$W.L..same_year = 100 * player_with_wins_combined$W.L..same_year
player_with_wins_combined$W.L..next_year = 100 * player_with_wins_combined$W.L..next_year

drop_cols = c("long_Tm.pitch", "Rk.bat", "G.bat", "long_Tm.bat", "Rk.pitch", "W", "L", "G.pitch", "long_
              "Age", "GS", "CG", "GS.field", "CG.field", "Rdrs", "Rdrs.yr", "Rgood")
player_with_wins_combined = player_with_wins_combined[, !(names(player_with_wins_combined) %in% drop_col

n.rows = nrow(player_with_wins_combined)
n.train = 0.8 * n.rows
train.rows = sample(n.rows, n.train)
train.df = player_with_wins_combined[train.rows,]
colnames(train.df)[colnames(train.df) == 'OPS.'] <- 'OPSplus'
colnames(train.df)[colnames(train.df) == 'ERA.'] <- 'ERAplus'
test.df = player_with_wins_combined[-train.rows,]
colnames(test.df)[colnames(test.df) == 'OPS.'] <- 'OPSplus'
colnames(test.df)[colnames(test.df) == 'ERA.'] <- 'ERAplus'
```

```r
# train.df
names(train.df)
```

```
##  [1] "Tm"              "Age.bat"         "PA"              "AB"
##  [5] "R.bat"           "H.bat"           "X2B"             "X3B"
##  [9] "HR.bat"          "RBI"             "SB"              "CS"
## [13] "BB.bat"          "SO.bat"          "BA"              "OBP"
## [17] "SLG"             "OPS"             "OPSplus"         "TB"
## [21] "GDP"             "HBP.bat"         "SH"              "SF"
## [25] "IBB.bat"         "year.bat"        "year_adj.bat"    "Age.pitch"
## [29] "W.L..same_year"  "ERA"             "GF"              "SHO"
## [33] "SV"              "IP"              "H.pitch"         "R.pitch"
## [37] "ER"              "HR.pitch"        "BB.pitch"        "IBB.pitch"
## [41] "SO.pitch"        "HBP.pitch"       "BK"              "WP"
## [45] "BF"              "ERAplus"         "FIP"             "WHIP"
## [49] "H9"              "HR9"             "BB9"             "SO9"
## [53] "SO.W"            "year.pitch"      "year_adj.pitch"  "Rk"
## [57] "G"               "Inn"             "Ch"              "PO"
## [61] "A"               "E"               "DP"              "Fld."
## [65] "Rtot"            "Rtot.yr"         "RF.9"            "RF.G"
## [69] "year"            "year_adj"        "W.L..next_year"  "tot_fa_war3"
## [73] "num_fas"
```

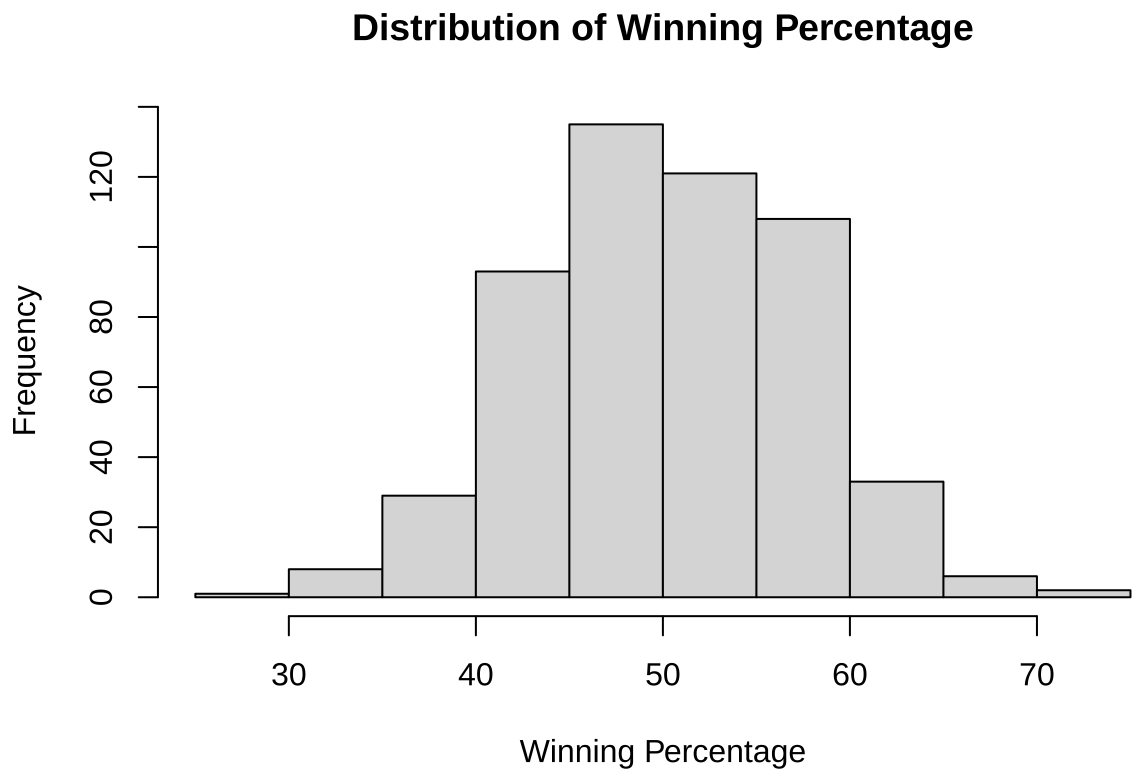
```r
# Explore Potential Predictors
par(mfrow=c(3,4))
plot(W.L..next_year ~ Age.bat, data=train.df,
     xlab="Batters' Avg. Age", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ Age.pitch, data=train.df,
     xlab="Pitchers' Avg. Age", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ BA, data=train.df,
     xlab="Batting Avg.", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ HR.bat, data=train.df,
     xlab="Avg. Home Runs", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ OPSplus, data=train.df,
     xlab="Adjusted OPS", ylab="Winning Percentage", cex=0.3)
plot(W.L..next_year ~ SO.W, data=train.df,
     xlab="Avg. Strikeout/Walk", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ HR9, data=train.df,
     xlab="Avg. HR/9", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ WHIP, data=train.df,
     xlab="WHIP", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ tot_fa_war3, data=train.df,
     xlab="Total Free Agent WAR3", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ E, data=train.df,
     xlab="Errors", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ Fld., data=train.df,
     xlab="Fielding %", ylab="Winning %", cex=0.3)
```




```r
# Summary statistics for winpct
summary(train.df$W.L..next_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   26.50   45.10   50.60   50.36   55.60   71.60
```

```r
# Histogram for winpct
hist(train.df$W.L..next_year, main="Distribution of Winning Percentage",
     xlab="Winning Percentage")
```




```r
# Correlation matrix
cor(train.df[, c("W.L..next_year", "Age.bat", "Age.pitch", "BA", "HR.bat", "OPS", "SO.W", "HR9", "WHIP"
```

```
##                W.L..next_year     Age.bat   Age.pitch          BA       HR.bat
## W.L..next_year     1.00000000  0.11752497  0.23987636  0.16874627  0.258708974
## Age.bat            0.11752497  1.00000000  0.51739922  0.18611215  0.145672229
## Age.pitch          0.23987636  0.51739922  1.00000000  0.12552375  0.197981229
## BA                 0.16874627  0.18611215  0.12552375  1.00000000  0.546066555
## HR.bat             0.25870897  0.14567223  0.19798123  0.54606655  1.000000000
## OPS                0.24261108  0.15356491  0.17370956  0.91233483  0.710686591
## SO.W               0.22686755 -0.09351711  0.01936649 -0.14304072  0.054107022
## HR9               -0.23045821 -0.16419856 -0.09426396  0.04534340  0.133878994
## WHIP              -0.33712072 -0.02977710 -0.10110815  0.16008020 -0.007431268
## tot_fa_war3        0.20721613  0.26224621  0.27976753  0.11804017  0.143102559
## E                  0.02741697  0.01968936  0.07974133  0.17313857  0.255708854
## Fld.               0.13427519  0.14194469  0.17685246  0.03896246  0.098147047
##                         OPS        SO.W         HR9         WHIP tot_fa_war3
## W.L..next_year  0.242611075  0.22686755 -0.23045821 -0.337120719  0.20721613
## Age.bat         0.153564912 -0.09351711 -0.16419856 -0.029777095  0.26224621
## Age.pitch       0.173709563  0.01936649 -0.09426396 -0.101108153  0.27976753
## BA              0.912334826 -0.14304072  0.04534340  0.160080201  0.11804017
## HR.bat          0.710686591  0.05410702  0.13387899 -0.007431268  0.14310256
## OPS             1.000000000 -0.02618238  0.18153117  0.115646503  0.16496218
## SO.W           -0.026182377  1.00000000 -0.06308450 -0.736753156  0.08645535
## HR9             0.181531167 -0.06308450  1.00000000  0.430177101 -0.01349860
## WHIP            0.115646503 -0.73675316  0.43017710  1.000000000 -0.06761557
```

```
## tot_fa_war3     0.164962175  0.08645535 -0.01349860 -0.067615573  1.00000000
## E               0.076113343 -0.35243217 -0.14754817  0.242938573 -0.06308137
## Fld.            0.009629124  0.03001657 -0.12599648 -0.159494533  0.07599568
##                          E        Fld.
## W.L..next_year  0.02741697  0.134275193
## Age.bat         0.01968936  0.141944693
## Age.pitch       0.07974133  0.176852463
## BA              0.17313857  0.038962460
## HR.bat          0.25570885  0.098147047
## OPS             0.07611334  0.009629124
## SO.W           -0.35243217  0.030016567
## HR9            -0.14754817 -0.125996475
## WHIP            0.24293857 -0.159494533
## tot_fa_war3    -0.06308137  0.075995676
## E               1.00000000 -0.076249214
## Fld.           -0.07624921  1.000000000
```

```
cor(train.df[, c("W.L..next_year", "Age.bat", "Age.pitch", "BA", "HR.bat", "OPS", "SO.W", "HR9", "WHIP"
```

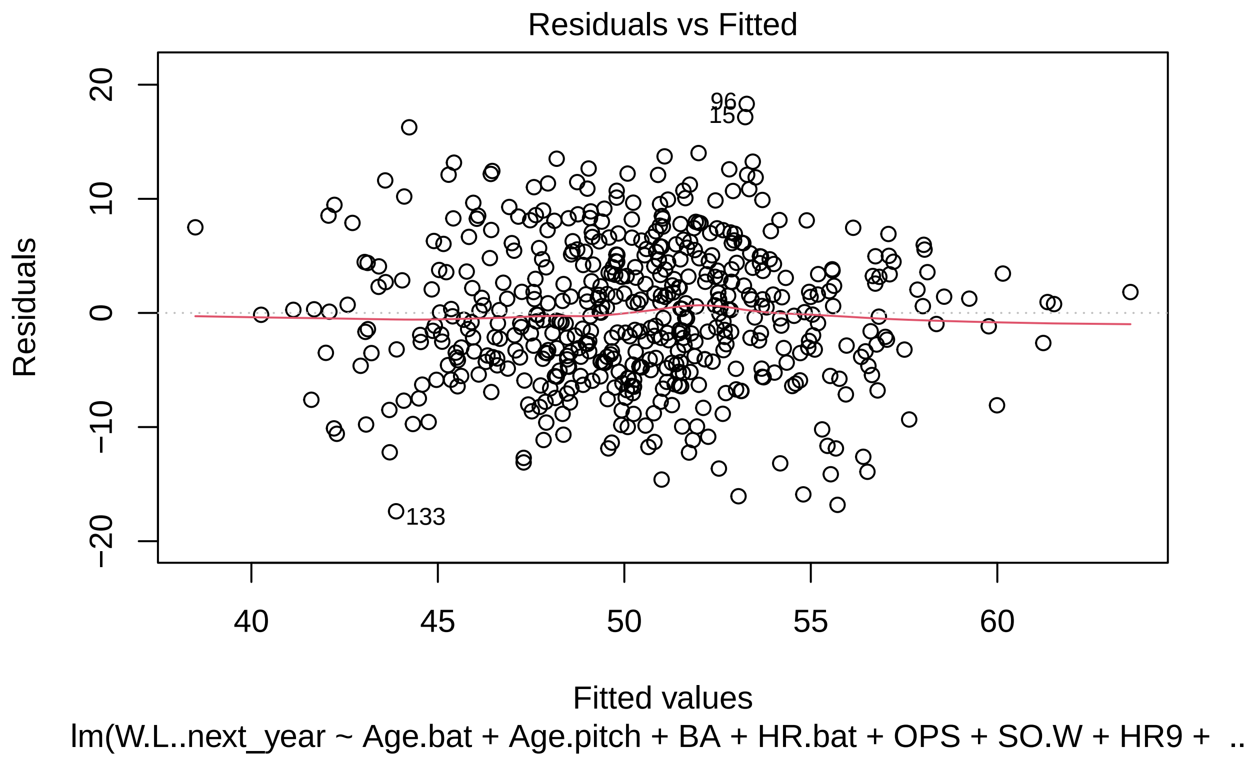
```
##                W.L..next_year      Age.bat   Age.pitch          BA       HR.bat
## W.L..next_year     1.00000000 0.0138121177 0.057540666 0.028475305 6.693033e-02
## Age.bat            0.01381212 1.0000000000 0.267701949 0.034637732 2.122040e-02
## Age.pitch          0.05754067 0.2677019489 1.000000000 0.015756213 3.919657e-02
## BA                 0.02847531 0.0346377324 0.015756213 1.000000000 2.981887e-01
## HR.bat             0.06693033 0.0212203982 0.039196567 0.298188682 1.000000e+00
## OPS                0.05886013 0.0235821823 0.030175012 0.832354834 5.050754e-01
## SO.W               0.05146888 0.0087454507 0.000375061 0.020460646 2.927570e-03
## HR9                0.05311098 0.0269611677 0.008885694 0.002056024 1.792359e-02
## WHIP               0.11365038 0.0008866754 0.010222859 0.025625671 5.522375e-05
## tot_fa_war3        0.04293852 0.0687730731 0.078269872 0.013933483 2.047834e-02
## E                  0.00075169 0.0003876710 0.006358680 0.029976964 6.538702e-02
## Fld.               0.01802983 0.0201482959 0.031276794 0.001518073 9.632843e-03
##                        OPS         SO.W          HR9         WHIP  tot_fa_war3
## W.L..next_year 5.886013e-02 0.0514688845 0.0531109847 1.136504e-01 0.0429385233
## Age.bat        2.358218e-02 0.0087454507 0.0269611677 8.866754e-04 0.0687730731
## Age.pitch      3.017501e-02 0.0003750610 0.0088856940 1.022286e-02 0.0782698721
## BA             8.323548e-01 0.0204606462 0.0020560240 2.562567e-02 0.0139334829
## HR.bat         5.050754e-01 0.0029275698 0.0179235850 5.522375e-05 0.0204783423
## OPS            1.000000e+00 0.0006855169 0.0329535647 1.337411e-02 0.0272125192
## SO.W           6.855169e-04 1.0000000000 0.0039796540 5.428052e-01 0.0074745276
## HR9            3.295356e-02 0.0039796540 1.0000000000 1.850523e-01 0.0001822121
## WHIP           1.337411e-02 0.5428052136 0.1850523384 1.000000e+00 0.0045718657
## tot_fa_war3    2.721252e-02 0.0074745276 0.0001822121 4.571866e-03 1.0000000000
## E              5.793241e-03 0.1242084357 0.0217704625 5.901915e-02 0.0039792599
## Fld.           9.272003e-05 0.0009009943 0.0158751118 2.543851e-02 0.0057753428
##                          E         Fld.
## W.L..next_year 0.000751690 1.802983e-02
## Age.bat        0.000387671 2.014830e-02
## Age.pitch      0.006358680 3.127679e-02
## BA             0.029976964 1.518073e-03
## HR.bat         0.065387018 9.632843e-03
## OPS            0.005793241 9.272003e-05
## SO.W           0.124208436 9.009943e-04
## HR9            0.021770462 1.587511e-02
```

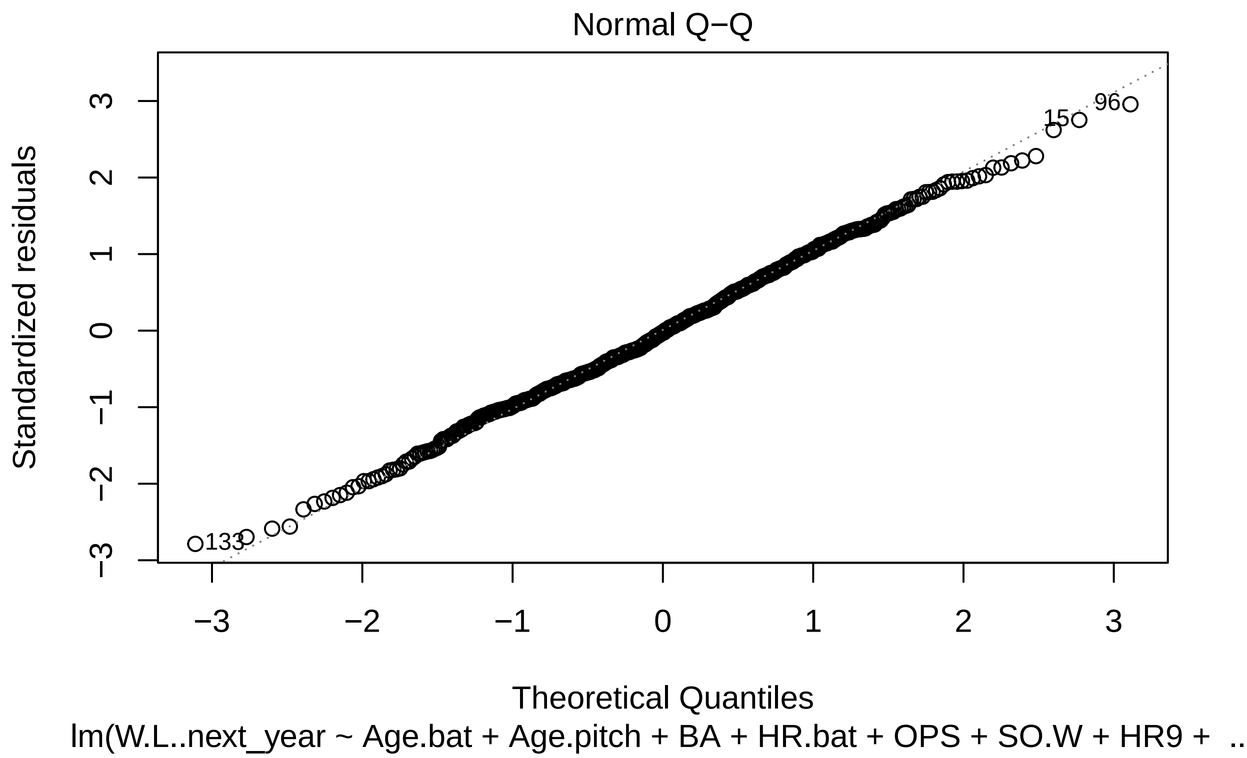
```
## WHIP           0.059019150 2.543851e-02
## tot_fa_war3    0.003979260 5.775343e-03
## E              1.000000000 5.813943e-03
## Fld.           0.005813943 1.000000e+00
```

```r
# Baseline Multiple Regression Model
baseline <- lm(W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat +
                 OPS + SO.W + HR9 + WHIP + tot_fa_war3 + E + Fld. , data = train.df)
summary(baseline)
```

```
##
## Call:
## lm(formula = W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat +
##     OPS + SO.W + HR9 + WHIP + tot_fa_war3 + E + Fld., data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.3800  -4.2722  -0.1538   4.3981  18.3189
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -10.25614   39.36219  -0.261 0.794536
## Age.bat       -0.23314    0.26826  -0.869 0.385202
## Age.pitch      0.62222    0.26487   2.349 0.019189 *
## BA          -139.86386   41.80505  -3.346 0.000880 ***
## HR.bat        -0.04254    0.13553  -0.314 0.753759
## OPS           76.68698   16.81169   4.562 6.33e-06 ***
## SO.W          -0.06723    0.81235  -0.083 0.934071
## HR9           -6.61423    1.77749  -3.721 0.000220 ***
## WHIP         -18.85895    4.85443  -3.885 0.000115 ***
## tot_fa_war3    0.06583    0.02349   2.802 0.005264 **
## E              0.54056    0.27910   1.937 0.053311 .
## Fld.          61.87723   38.16033   1.622 0.105511
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.281 on 524 degrees of freedom
## Multiple R-squared:  0.271,  Adjusted R-squared:  0.2557
## F-statistic: 17.71 on 11 and 524 DF,  p-value: < 2.2e-16
```

```r
# Asssess Linear Model Assumptions
plot(baseline, which=c(1,2))
```


Residuals vs Fitted
Residuals
Fitted values
lm(W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat + OPS + SO.W + HR9 + ..
96
15
133
20
10
0
−10
−20
40
45
50
55
60




```
RMSE <- function(y,yhat){
  SSE = sum((y-yhat)^2)
  SST = sum((y - mean(y))^2)
  return(sqrt(SSE/length(y)))
}

R2 <- function(y,yhat) {
  SSE = sum((y-yhat)^2)
  SST = sum((y-mean(y))^2)
  r.squared <- 1 - (SSE / SST)
  return(r.squared)
}
baseline.trainRMSE = RMSE(train.df$W.L..next_year, predict(baseline, newdata=train.df))
baseline.testRMSE = RMSE(test.df$W.L..next_year, predict(baseline, newdata=test.df))
baseline.trainR2 = R2(train.df$W.L..next_year, predict(baseline, newdata=train.df))
baseline.testR2 = R2(test.df$W.L..next_year, predict(baseline, newdata=test.df))
```

Linear Regression

```
colnames(train.df)
```

```
##  [1] "Tm"              "Age.bat"         "PA"              "AB"
##  [5] "R.bat"           "H.bat"           "X2B"             "X3B"
```

```
##  [9] "HR.bat"          "RBI"             "SB"              "CS"
## [13] "BB.bat"          "SO.bat"          "BA"              "OBP"
## [17] "SLG"             "OPS"             "OPSplus"         "TB"
## [21] "GDP"             "HBP.bat"         "SH"              "SF"
## [25] "IBB.bat"         "year.bat"        "year_adj.bat"    "Age.pitch"
## [29] "W.L..same_year"  "ERA"             "GF"              "SHO"
## [33] "SV"              "IP"              "H.pitch"         "R.pitch"
## [37] "ER"              "HR.pitch"        "BB.pitch"        "IBB.pitch"
## [41] "SO.pitch"        "HBP.pitch"       "BK"              "WP"
## [45] "BF"              "ERAplus"         "FIP"             "WHIP"
## [49] "H9"              "HR9"             "BB9"             "SO9"
## [53] "SO.W"            "year.pitch"      "year_adj.pitch"  "Rk"
## [57] "G"               "Inn"             "Ch"              "PO"
## [61] "A"               "E"               "DP"              "Fld."
## [65] "Rtot"            "Rtot.yr"         "RF.9"            "RF.G"
## [69] "year"            "year_adj"        "W.L..next_year"  "tot_fa_war3"
## [73] "num_fas"
```

```
# full linear regression models

# ignore Rk.bat, R.bat, RBI, year.bat, year_adj.bat, W, L, R.pitch, year.pitch
# year_adj.pitch, Rk, WL..next_year, year, year_adj, ERA, ERAplus
# Rtot, Rtot.yr, Rdrs, Rgood (hard to interpret)
lm.full <- lm(W.L..next_year ~ Age.bat + PA + AB + H.bat + X2B + X3B +
                HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                GF + SHO + SV + IP + H.pitch + HR.pitch +
                BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                G + Inn + Ch + PO + A + E + DP + Fld. +
                RF.9 + RF.G + tot_fa_war3 + num_fas,
              data = train.df)
lmfull.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.full, newdata=train.df))
```

```
## Warning in predict.lm(lm.full, newdata = train.df): prediction from a rank-
## deficient fit may be misleading
```

```
lmfull.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.full, newdata=test.df))
```

```
## Warning in predict.lm(lm.full, newdata = test.df): prediction from a rank-
## deficient fit may be misleading
```

```
lmfull.trainR2 = R2(train.df$W.L..next_year, predict(lm.full, newdata=train.df))
```

```
## Warning in predict.lm(lm.full, newdata = train.df): prediction from a rank-
## deficient fit may be misleading
```

```
lmfull.testR2 = R2(test.df$W.L..next_year, predict(lm.full, newdata=test.df))
```

```
## Warning in predict.lm(lm.full, newdata = test.df): prediction from a rank-
## deficient fit may be misleading
```

```r
lm.fullinteraction <- lm(W.L..next_year ~ (Age.bat + PA + AB + H.bat + X2B + X3B +
                HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                GF + SHO + SV + IP + H.pitch + HR.pitch +
                BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                G + Inn + Ch + PO + A + E + DP + Fld. +
                RF.9 + RF.G + tot_fa_war3 + num_fas)^2, data = train.df)
lmfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.fullinteraction, newdata=train.d
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = train.df): prediction from a
## rank-deficient fit may be misleading
```

```r
lmfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.fullinteraction, newdata=test.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = test.df): prediction from a
## rank-deficient fit may be misleading
```

```r
lmfullinteraction.trainR2 = R2(train.df$W.L..next_year, predict(lm.fullinteraction, newdata=train.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = train.df): prediction from a
## rank-deficient fit may be misleading
```

```r
lmfullinteraction.testR2 = R2(test.df$W.L..next_year, predict(lm.fullinteraction, newdata=test.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = test.df): prediction from a
## rank-deficient fit may be misleading
```

```r
# Ridge Regression
set.seed(139)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-4
```

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked _by_ '.GlobalEnv':
##
##     R2, RMSE
```

```r
# regularize full model
X.full = model.matrix(lm.full)[,-1] # drop intercept
best_lambda = cv.glmnet(X.full, train.df$W.L..next_year, alpha=0, lambda=10^seq(-4, 4, 0.1))$lambda.min
ridges.full = glmnet(X.full, train.df$W.L..next_year, alpha=0,
                     lambda=best_lambda)
varImp(ridges.full, lambda=best_lambda)
```

```
##                   Overall
## Age.bat        5.752159e-01
## PA             4.169554e-03
## AB             7.168485e-03
## H.bat          3.168249e-05
## X2B            8.584614e-03
## X3B            1.109549e+00
## HR.bat         2.119815e-01
## SB             6.806400e-02
## CS             3.390557e-02
## BB.bat         1.414319e-01
## SO.bat         3.652799e-02
## BA             5.997285e+00
## OBP            2.554351e+01
## SLG            1.418188e+01
## OPS            9.939370e+00
## OPSplus        7.547341e-03
## TB             6.824692e-03
## GDP            2.307993e-01
## HBP.bat        2.386825e-01
## SH             8.981258e-02
## SF             8.497103e-02
## IBB.bat        2.708115e-01
## Age.pitch      4.154787e-01
## W.L..same_year 2.058143e-02
## GF             8.157071e-02
## SHO            6.247652e+00
## SV             3.289460e-01
## IP             4.275370e-03
## H.pitch        1.211208e-02
## HR.pitch       2.097323e-01
## BB.pitch       1.333564e-02
## IBB.pitch      1.673001e-01
## SO.pitch       2.530369e-02
## HBP.pitch      1.453254e-01
## BK             1.185275e+00
## WP             2.455001e-02
## BF             7.827181e-05
## FIP            6.498817e-01
## WHIP           5.841223e+00
## H9             1.214867e+00
## HR9            2.030656e+00
## BB9            8.721206e-02
## SO9            2.270999e-01
## SO.W           3.418090e-01
## G              2.707582e-02
```

```
## Inn            3.372902e-03
## Ch             2.487647e-04
## PO             1.125522e-03
## A              2.452944e-03
## E              3.082556e-01
## DP             7.332010e-02
## Fld.           4.450743e+01
## RF.9           2.287439e+00
## RF.G           2.650137e+00
## tot_fa_war3    7.077088e-02
## num_fas        1.333073e-01
```

```
X.full.test = model.matrix(lm.full, data=test.df)[,-1] # drop intercept

yhats.full.train = predict(ridges.full, X.full)
ridgesfull.trainRMSE = RMSE(train.df$W.L..next_year, yhats.full.train) # train RMSE
ridgesfull.trainR2 = R2(train.df$W.L..next_year, yhats.full.train) # train R2

yhats.full.test = predict(ridges.full, X.full.test)
#plot(RMSE.ridges.full.test~log(ridges.full$lambda, 10), type='l')
ridgesfull.testRMSE = RMSE(test.df$W.L..next_year, yhats.full.test) # test RMSE
ridgesfull.testR2 = R2(test.df$W.L..next_year, yhats.full.test) # test R2
```

```
set.seed(139)
# regularize full interaction model
X.fullinteraction = model.matrix(lm.fullinteraction)[,-1] # drop intercept

best_lambda = cv.glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=0,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min
ridges.fullinteraction = glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=0,
                     lambda=best_lambda)
X.fullinteraction.test = model.matrix(lm.fullinteraction, data=test.df)[,-1] # drop intercept

yhats.fullinteraction.train = predict(ridges.fullinteraction, X.fullinteraction)
ridgesfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, yhats.fullinteraction.train) # train RM
ridgesfullinteraction.trainR2 = R2(train.df$W.L..next_year, yhats.fullinteraction.train) # train R2

yhats.fullinteraction.test = predict(ridges.fullinteraction, X.fullinteraction.test)
#plot(RMSE.ridges.fullinteraction.test~log(ridges.fullinteraction$lambda, 10), type='l')
ridgesfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, yhats.fullinteraction.test) # train RMSE
ridgesfullinteraction.testR2 = R2(test.df$W.L..next_year, yhats.fullinteraction.test) # train R2
```

```
# Lasso Regression
# regularize full model
set.seed(139)
best_lambda = cv.glmnet(X.full, train.df$W.L..next_year, alpha=1,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min
lassos.full = glmnet(X.full, train.df$W.L..next_year, alpha=1,
                     lambda=best_lambda)
varImp(lassos.full, lambda=best_lambda)
```

```
##                    Overall
## Age.bat        0.760403704
```

```
## PA              0.000000000
## AB              0.013798186
## H.bat           0.000000000
## X2B             0.000000000
## X3B             1.138148047
## HR.bat          0.293888771
## SB              0.041199529
## CS              0.000000000
## BB.bat          0.215962524
## SO.bat          0.062850770
## BA              0.000000000
## OBP            10.863462973
## SLG             7.962457059
## OPS            17.605036197
## OPSplus         0.000000000
## TB              0.000000000
## GDP             0.207791848
## HBP.bat         0.248262694
## SH              0.000000000
## SF              0.000000000
## IBB.bat         0.114433745
## Age.pitch       0.406103170
## W.L..same_year  0.000000000
## GF              0.000000000
## SHO             5.947455726
## SV              0.248392153
## IP              0.000000000
## H.pitch         0.000000000
## HR.pitch        0.298752976
## BB.pitch        0.000000000
## IBB.pitch       0.071047986
## SO.pitch        0.031290182
## HBP.pitch       0.049407289
## BK              0.759943266
## WP              0.000000000
## BF              0.000000000
## FIP             0.758780126
## WHIP            5.875680328
## H9              1.601681602
## HR9             1.505850813
## BB9             0.000000000
## SO9             0.066333113
## SO.W            0.183684122
## G               0.000000000
## Inn             0.007817371
## Ch              0.000000000
## PO              0.000000000
## A               0.000000000
## E               0.245624203
## DP              0.070496142
## Fld.           37.971227099
## RF.9            3.210209394
## RF.G            3.901945224
## tot_fa_war3     0.079787301
```

```
## num_fas         0.132404107
```

```
yhats.full.train = predict(lassos.full, X.full)
lassosfull.trainRMSE = RMSE(train.df$W.L..next_year, yhats.full.train) # train RMSE
lassosfull.trainR2 = R2(train.df$W.L..next_year, yhats.full.train) # train R2

yhats.full.test = predict(lassos.full, X.full.test)
#plot(RMSE.lassos.full.test~log(ridges.full$lambda, 10), type='l')
lassosfull.testRMSE = RMSE(test.df$W.L..next_year, yhats.full.test) # test RMSE
lassosfull.testR2 = R2(test.df$W.L..next_year, yhats.full.test) # test RMSE
```

```
# regularize full interaction model
set.seed(139)
best_lambda = cv.glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=1,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min
lassos.fullinteraction = glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=1,
                     lambda=best_lambda)
yhats.fullinteraction.train = predict(lassos.fullinteraction, X.fullinteraction)
lassosfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, yhats.fullinteraction.train) # train RM
lassosfullinteraction.trainR2 = R2(train.df$W.L..next_year, yhats.fullinteraction.train) # train R2

yhats.fullinteraction.test = predict(lassos.fullinteraction, X.fullinteraction.test)
#plot(RMSE.lassos.fullinteraction.test~log(lassos.fullinteraction$lambda, 10), type='l')
lassosfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, yhats.fullinteraction.test) # train RMSE
lassosfullinteraction.testR2 = R2(test.df$W.L..next_year, yhats.fullinteraction.test) # train R2
```

```
# Stepwise
lm.step = step(lm.full, scope=c(lower=formula(W.L..next_year~1),
                                upper=lm.fullinteraction), trace=0, direction="both")
formula(lm.step)
```

```
## W.L..next_year ~ Age.bat + PA + AB + H.bat + X3B + OPS + HBP.bat +
##     Age.pitch + SHO + SO.pitch + BF + H9 + HR9 + BB9 + Inn +
##     RF.9 + RF.G + tot_fa_war3 + num_fas
```

```
lmstep.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.step, newdata=train.df))
lmstep.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.step, newdata=test.df))
lmstep.trainR2 = R2(train.df$W.L..next_year, predict(lm.step, newdata=train.df))
lmstep.testR2 = R2(test.df$W.L..next_year, predict(lm.step, newdata=test.df))
```

```
# model comparison
RMSE.df = data.frame(trainRMSE = c(baseline.trainRMSE,
                                   lmfull.trainRMSE,
                                   lmfullinteraction.trainRMSE,
                                   ridgesfull.trainRMSE,
                                   ridgesfullinteraction.trainRMSE,
                                   lassosfull.trainRMSE,
                                   lassosfullinteraction.trainRMSE,
                                   lmstep.trainRMSE),
                     testRMSE = c(baseline.testRMSE,
                                   lmfull.testRMSE,
                                   lmfullinteraction.testRMSE,
```

```
                                   ridgesfull.testRMSE,
                                   ridgesfullinteraction.testRMSE,
                                   lassosfull.testRMSE,
                                   lassosfullinteraction.testRMSE,
                                   lmstep.testRMSE),
                     trainR2 = c(baseline.trainR2,
                                   lmfull.trainR2,
                                   lmfullinteraction.trainR2,
                                   ridgesfull.trainR2,
                                   ridgesfullinteraction.trainR2,
                                   lassosfull.trainR2,
                                   lassosfullinteraction.trainR2,
                                   lmstep.trainR2),
                     testR2 = c(baseline.testR2,
                                   lmfull.testR2,
                                   lmfullinteraction.testR2,
                                   ridgesfull.testR2,
                                   ridgesfullinteraction.testR2,
                                   lassosfull.testR2,
                                   lassosfullinteraction.testR2,
                                   lmstep.testR2))
rownames(RMSE.df) <- c("baseline", "full", "full interaction",
                       "ridge full", "ridge full interaction",
                       "lasso full", "lasso full interaction",
                       "step")
RMSE.df
```

```
##                           trainRMSE     testRMSE   trainR2        testR2
## baseline               6.210565e+00     7.326775 0.2709964  2.591406e-01
## full                   5.660738e+00     7.068803 0.3943615  3.103929e-01
## full interaction       6.096283e-07 58781.849584 1.0000000 -4.768662e+07
## ridge full             5.764690e+00     7.079809 0.3719137  3.082437e-01
## ridge full interaction 5.721998e+00     6.991096 0.3811822  3.254711e-01
## lasso full             5.737253e+00     7.066628 0.3778781  3.108171e-01
## lasso full interaction 5.656210e+00     7.119921 0.3953298  3.003830e-01
## step                   5.764326e+00     7.075806 0.3719930  3.090257e-01
```

Decision Tree/Random Forest

```
set.seed(139)
library(rpart)

RMSE = function(y,yhat){
  return(sqrt(mean((y-yhat)^2)))
}

test.df = subset(test.df, test.df$Tm != 'CLE')
tree1 = rpart(formula(lm.full),data=train.df, control = list(minsplit=1,cp=0,maxdepth=20))
yhat.tree1.train = predict(tree1)
yhat.tree1.test = predict(tree1, newdata = test.df)
RMSE.tree1.train = RMSE(train.df$W.L..next_year,yhat.tree1.train)
```

```
RMSE.tree1.test = RMSE(test.df$W.L..next_year,yhat.tree1.test)
data.frame(train=RMSE.tree1.train,test=RMSE.tree1.test)
```

```
##      train     test
## 1 3.879832 8.406397
```

```
set.seed(139)
best.cp = tree1$cptable[,"CP"][which.min(tree1$cptable[,"xerror"])]
tree2 = prune(tree1,best.cp)
yhat.tree2.train = predict(tree2)
yhat.tree2.test = predict(tree2,newdata=test.df)
RMSE.tree2.train = RMSE(train.df$W.L..next_year,yhat.tree2.train)
RMSE.tree2.test = RMSE(test.df$W.L..next_year,yhat.tree2.test)
data.frame(train=RMSE.tree2.train,test=RMSE.tree2.test)
```

```
##      train     test
## 1 6.226519 7.778095
```

```
set.seed(139)
tree3 = rpart(W.L..next_year~W.L..same_year + Age.pitch + WHIP,
              data=train.df, control = list(minsplit=1, cp=0, maxdepth=20))
yhat.tree3.train = predict(tree3)
yhat.tree3.test = predict(tree3, newdata = test.df)
RMSE.tree3.train = RMSE(train.df$W.L..next_year,yhat.tree3.train)
RMSE.tree3.test = RMSE(test.df$W.L..next_year,yhat.tree3.test)
data.frame(train=RMSE.tree3.train,test=RMSE.tree3.test)
```

```
##      train     test
## 1 5.259777 8.073087
```

```
set.seed(139)
best.cp = tree3$cptable[,"CP"][which.min(tree3$cptable[,"xerror"])]
tree4 = prune(tree3,best.cp)
yhat.tree4.train = predict(tree4)
yhat.tree4.test = predict(tree4,newdata=test.df)
RMSE.tree4.train = RMSE(train.df$W.L..next_year,yhat.tree4.train)
RMSE.tree4.test = RMSE(test.df$W.L..next_year,yhat.tree4.test)
data.frame(train=RMSE.tree4.train,test=RMSE.tree4.test)
```

```
##      train     test
## 1 6.669579 7.866976
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
set.seed(139)
maxnodes = c(100,200,500)
ntree= 200
rmses.bag = rep(NA,length(maxnodes))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:length(maxnodes)){
  bagtemp = randomForest(formula(lm.full),data=train.df,
                        mtry=56, maxnodes=maxnodes[i], ntree=ntree)
  rmses.bag[i]=RMSE(train.df$W.L..next_year, bagtemp$predicted)
  if(rmses.bag[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=rmses.bag[i]
    bag=bagtemp
  }
}
data.frame(maxnodes=maxnodes, RMSE=rmses.bag)
```

```
##   maxnodes     RMSE
## 1      100 6.472173
## 2      200 6.359278
## 3      500 6.401620
```

```
yhat.bag.train = predict(bag)
yhat.bag.test = predict(bag, newdata = test.df)
RMSE.bag.train = RMSE(train.df$W.L..next_year,yhat.bag.train)
RMSE.bag.test = RMSE(test.df$W.L..next_year,yhat.bag.test)
data.frame(train=RMSE.bag.train,test=RMSE.bag.test)
```

```
##      train    test
## 1 6.359278 7.44455
```

```
library(randomForest)
set.seed(139)
maxnodes = c(100,200,500)
mtry = c(15, 25, 35, 45, 55)
ntree=200
pars = expand.grid(maxnodes=maxnodes,mtry=mtry)
RMSEs = rep(NA,nrow(pars))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:nrow(pars)){
  rftemp = randomForest(formula(lm.full),data=train.df,
                        mtry=pars$mtry[i], maxnodes=pars$maxnodes[i], ntree=ntree)
```

```r
  RMSEs[i]=RMSE(train.df$W.L..next_year, rftemp$predicted)
  if(RMSEs[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=RMSEs[i]
    rf1=rftemp
  }
}
data.frame(maxnodes=pars$maxnodes,mtry=pars$mtry,RMSE=RMSEs)
```

```
##    maxnodes mtry     RMSE
## 1       100   15 6.412723
## 2       200   15 6.338991
## 3       500   15 6.357269
## 4       100   25 6.359432
## 5       200   25 6.371718
## 6       500   25 6.361043
## 7       100   35 6.423718
## 8       200   35 6.420993
## 9       500   35 6.400056
## 10      100   45 6.422853
## 11      200   45 6.394741
## 12      500   45 6.425398
## 13      100   55 6.434080
## 14      200   55 6.455467
## 15      500   55 6.388878
```

```r
pars[which(RMSEs==bestRMSE),]
```

```
##   maxnodes mtry
## 2      200   15
```

```r
yhat.rf1.train = predict(rf1)
yhat.rf1.test = predict(rf1, newdata = test.df)
RMSE.rf1.train = RMSE(train.df$W.L..next_year,yhat.rf1.train)
RMSE.rf1.test = RMSE(test.df$W.L..next_year,yhat.rf1.test)
data.frame(train=RMSE.tree1.train,test=RMSE.rf1.test)
```

```
##      train     test
## 1 3.879832 7.451836
```

```r
importance(rf1)
```

```
##                 IncNodePurity
## Age.bat              398.4691
## PA                   195.4966
## AB                   277.5608
## H.bat                182.8005
## X2B                  344.1605
## X3B                  462.1844
## HR.bat               597.0227
## SB                   339.0336
```

```
## CS                  335.9434
## BB.bat              852.8076
## SO.bat              273.7432
## BA                  345.0928
## OBP                 720.1636
## SLG                 507.0741
## OPS                 585.1021
## OPSplus             583.9218
## TB                  202.3875
## GDP                 389.0385
## HBP.bat             391.1010
## SH                  362.8479
## SF                  332.2869
## IBB.bat             455.8381
## Age.pitch           722.7145
## W.L..same_year     1567.6407
## GF                  240.0901
## SHO                 402.4193
## SV                  873.7227
## IP                  475.8722
## H.pitch             333.2561
## HR.pitch            361.4512
## BB.pitch            287.0998
## IBB.pitch           377.2656
## SO.pitch           1106.5547
## HBP.pitch           334.5743
## BK                  531.9950
## WP                  379.8389
## BF                  372.3941
## FIP                 872.2285
## WHIP               1244.4283
## H9                 1384.2177
## HR9                 612.7717
## BB9                 427.3121
## SO9                 392.7926
## SO.W                382.4849
## G                   536.2027
## Inn                 304.1023
## Ch                  354.4777
## PO                  329.7342
## A                   293.8184
## E                   409.1964
## DP                  298.7246
## Fld.                431.3260
## RF.9                395.7701
## RF.G                374.6570
## tot_fa_war3         792.5348
## num_fas             336.3583
```

```
library(randomForest)
set.seed(139)
maxnodes = c(100,200,500)
mtry = c(1,2,3)
ntree=200
```

```r
pars = expand.grid(maxnodes=maxnodes,mtry=mtry)
RMSEs = rep(NA,nrow(pars))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:nrow(pars)){
  rftemp = randomForest(W.L..next_year ~ W.L..same_year + Age.pitch + WHIP, data=train.df,
                        mtry=pars$mtry[i], maxnodes=pars$maxnodes[i], ntree=ntree)
  RMSEs[i]=RMSE(train.df$W.L..next_year, rftemp$predicted)
  if(RMSEs[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=RMSEs[i]
    rf2=rftemp
  }
}
data.frame(maxnodes=pars$maxnodes,mtry=pars$mtry,RMSE=RMSEs)
```

```
##   maxnodes mtry     RMSE
## 1      100    1 6.670368
## 2      200    1 6.705663
## 3      500    1 6.651213
## 4      100    2 6.668633
## 5      200    2 6.723444
## 6      500    2 6.751639
## 7      100    3 6.649184
## 8      200    3 6.726162
## 9      500    3 6.706977
```

```r
pars[which(RMSEs==bestRMSE),]
```

```
##   maxnodes mtry
## 7      100    3
```

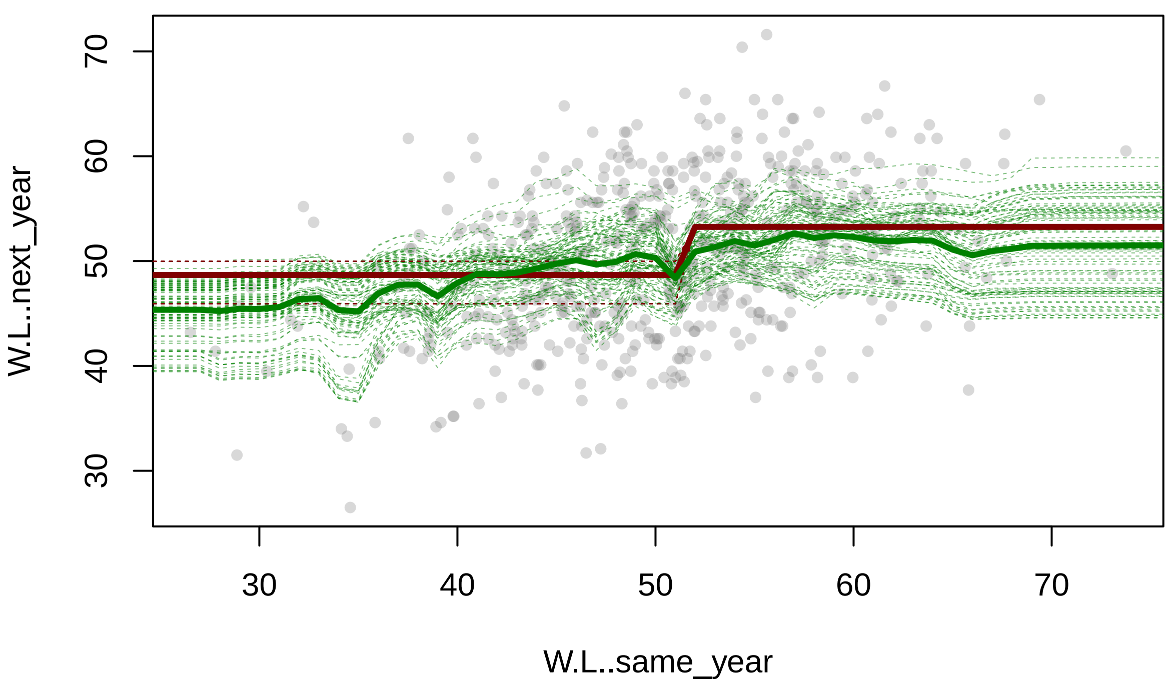
```r
yhat.rf2.train = predict(rf2)
yhat.rf2.test = predict(rf2, newdata = test.df)
RMSE.rf2.train = RMSE(train.df$W.L..next_year,yhat.rf2.train)
RMSE.rf2.test = RMSE(test.df$W.L..next_year,yhat.rf2.test)
data.frame(train=RMSE.tree1.train,test=RMSE.rf2.test)
```

```
##      train     test
## 1 3.879832 7.575087
```

```r
importance(rf2)
```

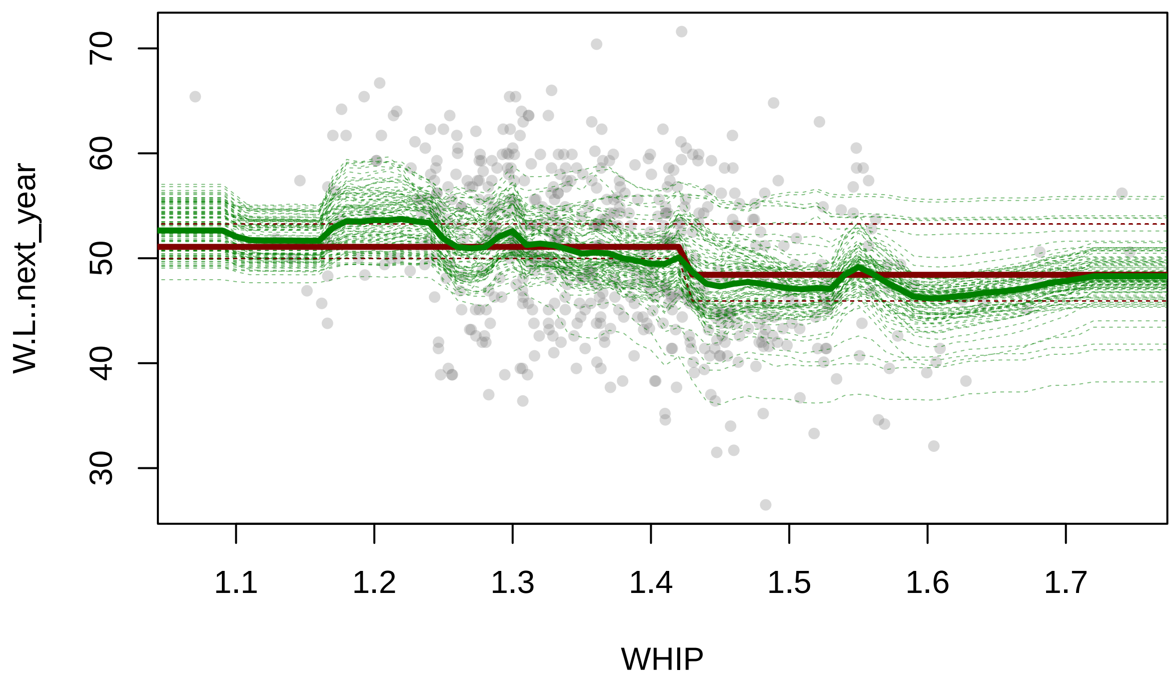
```
##                 IncNodePurity
## W.L..same_year       8539.024
## Age.pitch            6240.002
## WHIP                 7168.020
```

```
set.seed(139)
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(0,100,1)
plot(W.L..next_year~W.L..same_year, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$W.L..same_year=dummyx
  yhat = predict(tree4,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf2,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



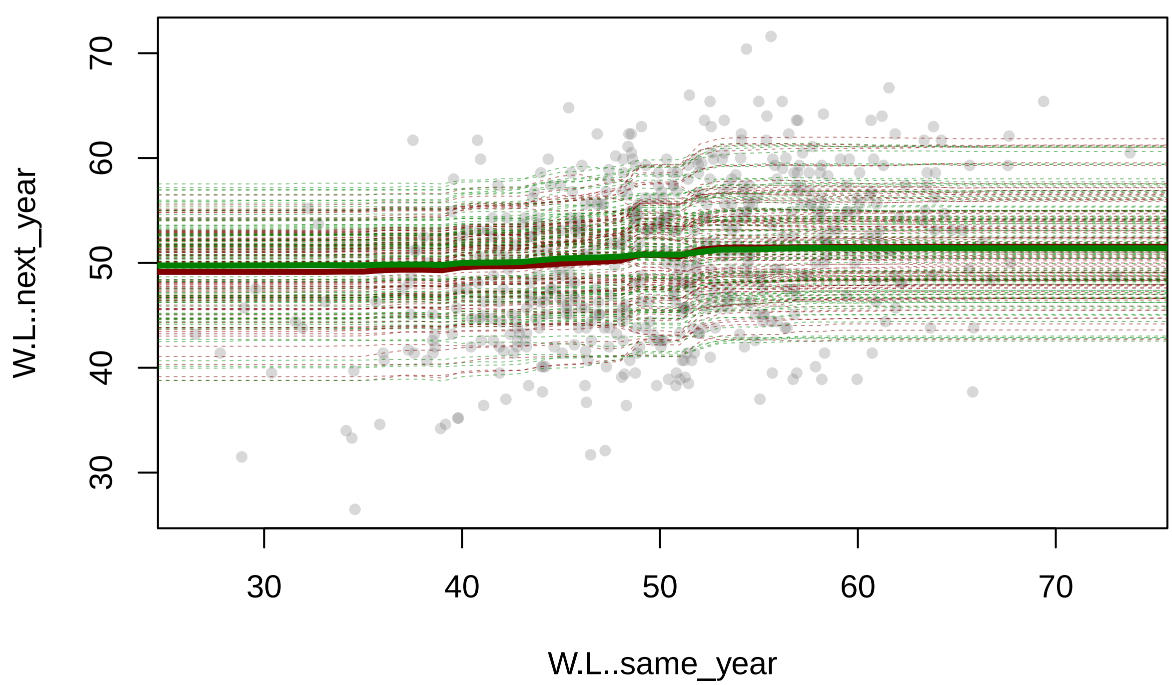

```
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(1,2,.01)
```

```r
plot(W.L..next_year~WHIP, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$WHIP=dummyx
  yhat = predict(tree4,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf2,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



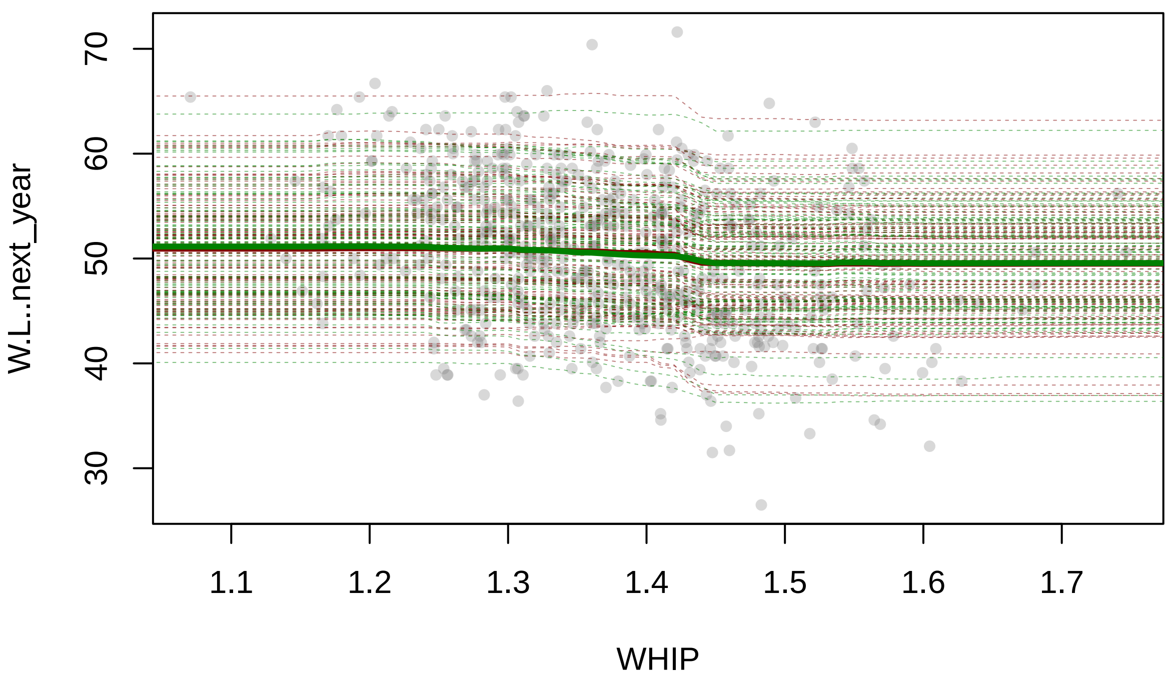

```r
set.seed(139)
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(0,100,1)
plot(W.L..next_year~W.L..same_year, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
```

```
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$W.L..same_year=dummyx
  yhat = predict(bag,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf1,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```

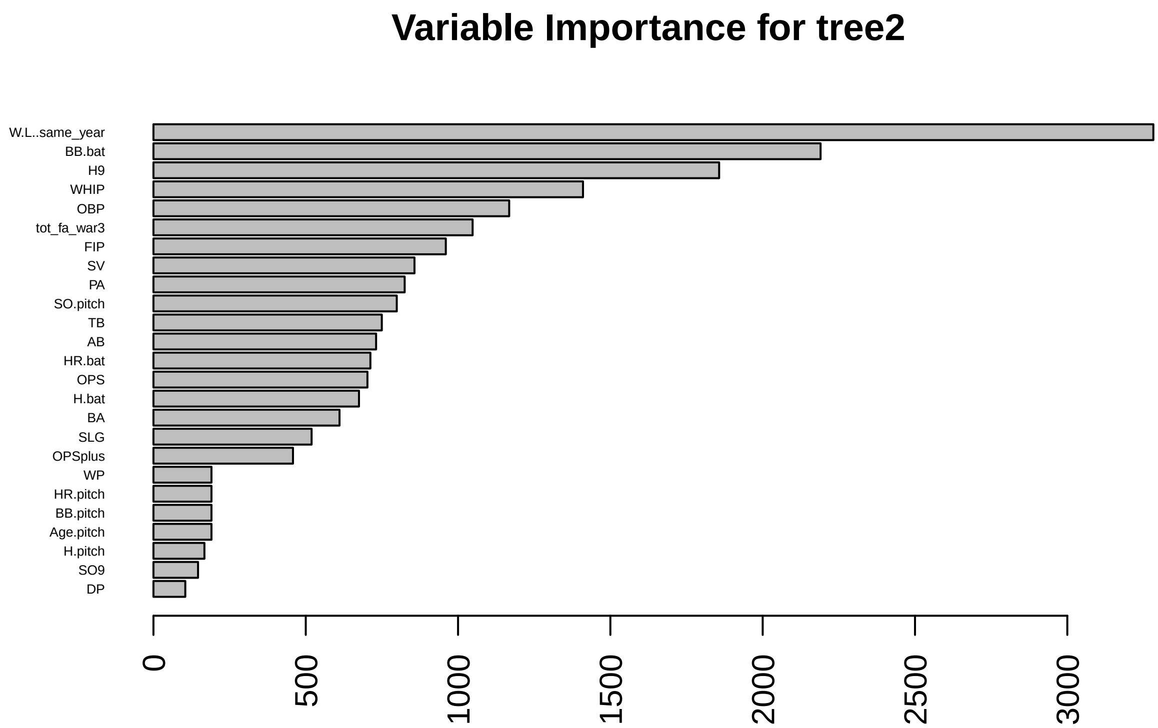



```
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(1,2,.01)
plot(W.L..next_year~WHIP, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$WHIP=dummyx
  yhat = predict(bag,new=rows)
```

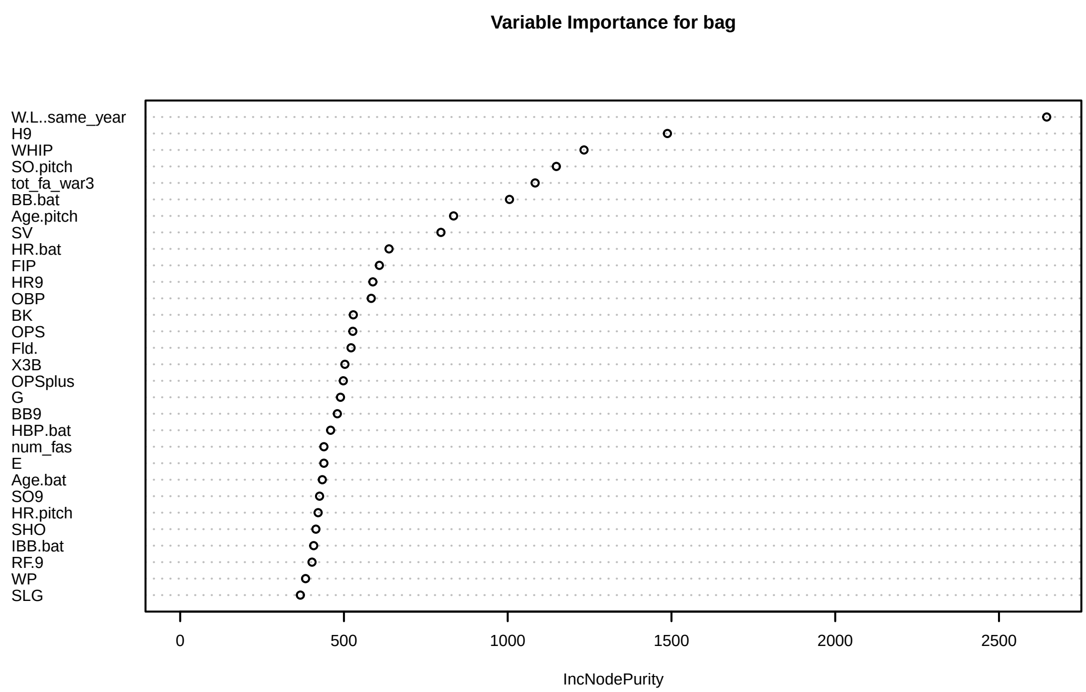
```
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf1,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



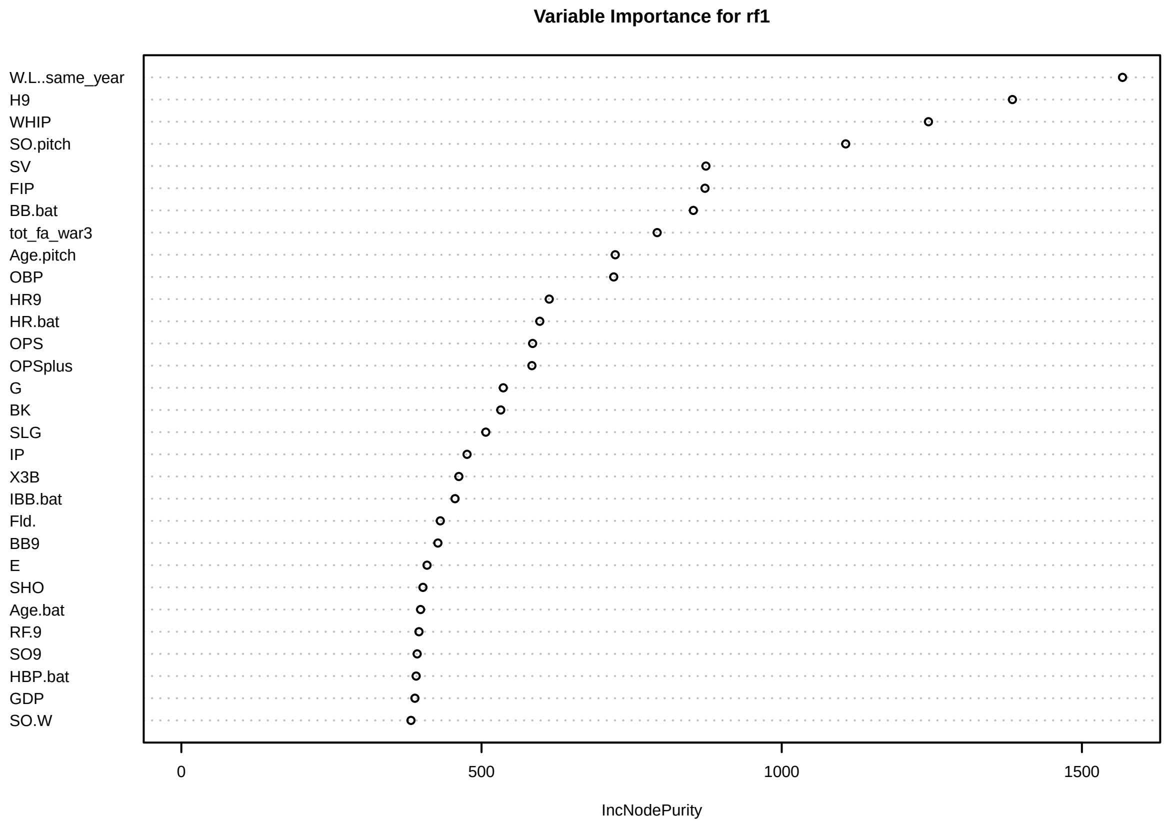

```
barplot(sort(tree2$variable.importance),horiz = T,las=2,cex.names = 0.4, main='Variable Importance for
```



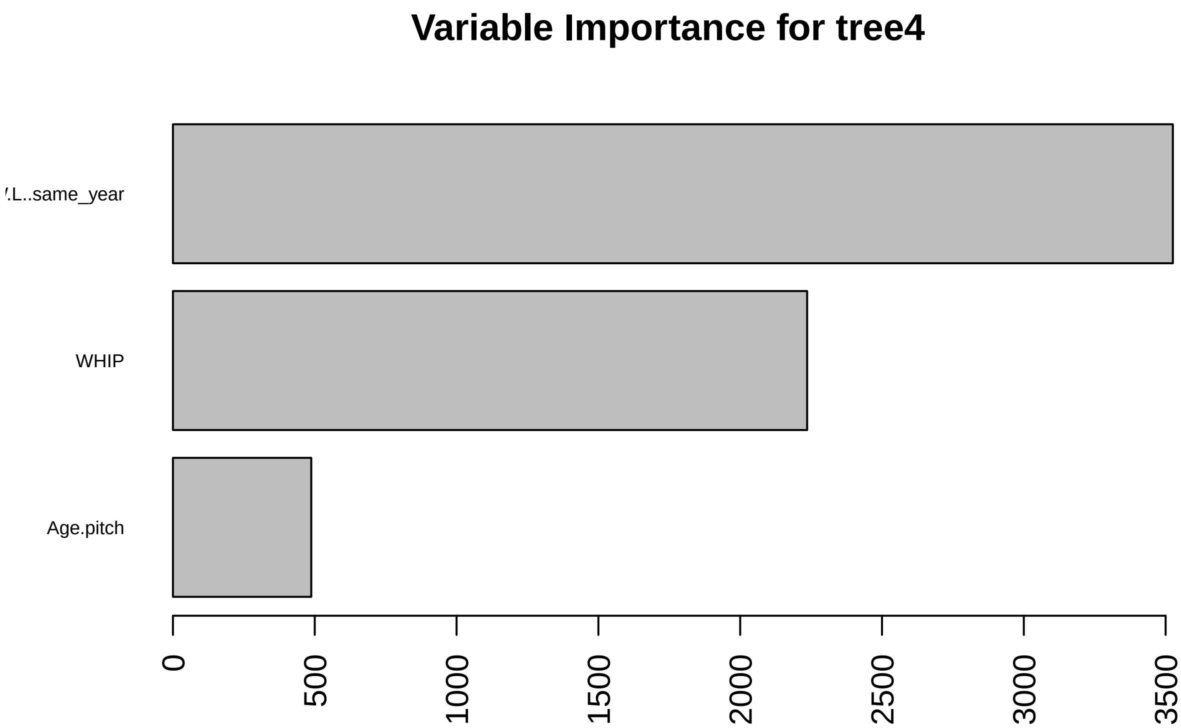

```
varImpPlot(bag, cex=0.5, main='Variable Importance for bag')
```



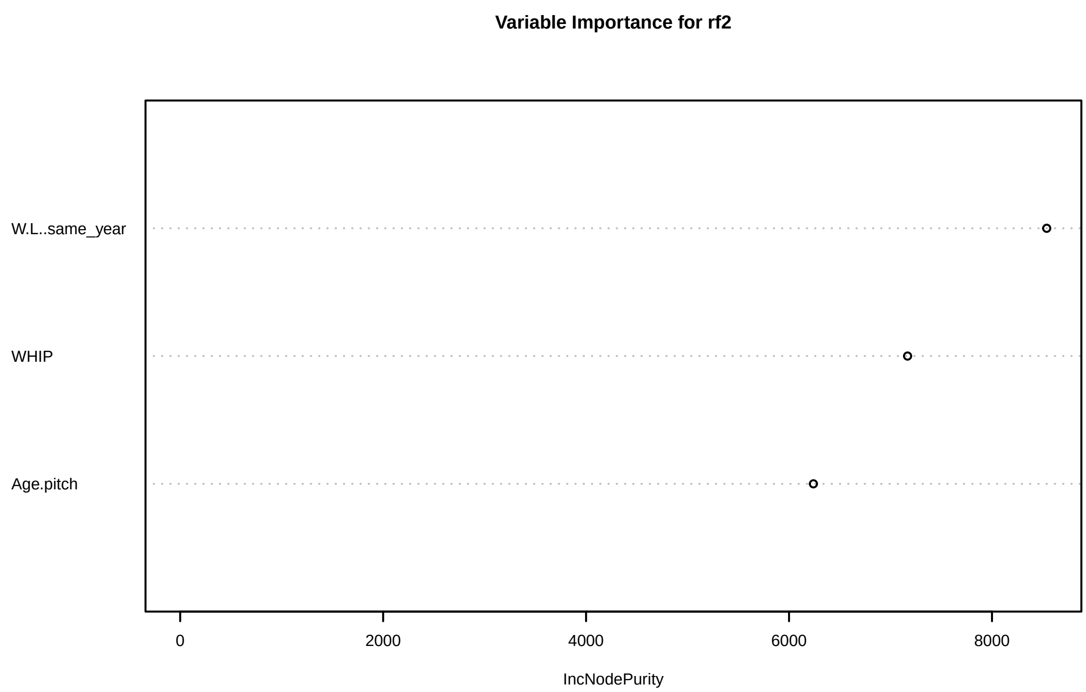

```
varImpPlot(rf1,cex=0.5, main='Variable Importance for rf1')
```




```
barplot(sort(tree4$variable.importance),horiz = T,las=2,cex.names = 0.6, main='Variable Importance for t
```




```r
varImpPlot(rf2, cex=0.5, main='Variable Importance for rf2')
```




```r
tab <- matrix(c(RMSE.tree1.train, RMSE.tree1.test,
  RMSE.tree2.train, RMSE.tree2.test,
  RMSE.bag.train, RMSE.bag.test,
  RMSE.rf1.train, RMSE.rf1.test,
  RMSE.rf2.train, RMSE.rf2.test,
  RMSE.tree4.train, RMSE.tree4.test), nrow=6, byrow = TRUE
)
colnames(tab) <- c('train','test')
rownames(tab) <- c('tree1','tree2','bag', 'rf1', 'rf2', 'tree4')
tab <- as.table(tab)
tab
```

```
##          train     test
## tree1 3.879832 8.406397
## tree2 6.226519 7.778095
## bag   6.359278 7.444550
## rf1   6.338991 7.451836
## rf2   6.649184 7.575087
## tree4 6.669579 7.866976
```

```r
library(lme4)
set.seed(139)

# for (i in 1997:2022){
#   lmer_model <- lmer(team_data[[i]]$W.L.~poly(team_data[[i]]$BatAge, 2, raw = TRUE) + (1 + poly(team_
```

```
#   summary(lmer_model)
# }

lmer_model <- lmer(train.df$W.L..next_year ~ poly(train.df$Age.bat, 2, raw = FALSE) + (1 + poly(train.d
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$Age.bat, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$Age.bat, 2, raw = FALSE) |
##         train.df$Tm))
##
## REML criterion at convergence: 3562.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1945 -0.7110  0.0377  0.6549  3.3518
##
## Random effects:
##  Groups        Name                                    Variance Std.Dev. Corr
##  train.df.Tm   (Intercept)                              11.27    3.357
##  train.df.Tm.1 poly(train.df$Age.bat, 2, raw = FALSE)1 299.31   17.300
##                poly(train.df$Age.bat, 2, raw = FALSE)2 153.86   12.404   -1.00
##  Residual                                               41.61    6.451
## Number of obs: 536, groups:  train.df$Tm, 29
##
## Fixed effects:
##                                         Estimate Std. Error t value
## (Intercept)                              50.1414     0.6908  72.585
## poly(train.df$Age.bat, 2, raw = FALSE)1  -3.3801     8.2867  -0.408
## poly(train.df$Age.bat, 2, raw = FALSE)2   4.0653     7.6890   0.529
##
## Correlation of Fixed Effects:
##                     (Intr) p(.$A.,2,r=FALSE)1
## p(.$A.,2,r=FALSE)1 -0.003
## p(.$A.,2,r=FALSE)2  0.028 -0.134
```

```
lmer_model <- lmer(train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) + (1 + poly(train.df$BA,
```

```
## boundary (singular) fit: see help('isSingular')
```

```
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$BA, 2, raw = FALSE) |
##         train.df$Tm))
##
## REML criterion at convergence: 3545.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.0813 -0.6780 -0.0020  0.6808  3.1719
##
## Random effects:
##  Groups        Name                              Variance  Std.Dev.  Corr
##  train.df.Tm   (Intercept)                       1.137e+01 3.3716324
##  train.df.Tm.1 poly(train.df$BA, 2, raw = FALSE)1 1.858e-07 0.0004311
##                poly(train.df$BA, 2, raw = FALSE)2 1.262e-08 0.0001124 1.00
##  Residual                                        4.083e+01 6.3895753
## Number of obs: 536, groups:  train.df$Tm, 29
##
## Fixed effects:
##                                   Estimate Std. Error t value
## (Intercept)                        50.1940     0.6892  72.827
## poly(train.df$BA, 2, raw = FALSE)1  35.8076     8.4294   4.248
## poly(train.df$BA, 2, raw = FALSE)2   6.7245     6.7529   0.996
##
## Correlation of Fixed Effects:
##                    (Intr) p(.$BA,2,r=FALSE)1
## p(.$BA,2,r=FALSE)1 0.010
## p(.$BA,2,r=FALSE)2 0.005  0.016
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```r
# lmer_model <- lmer(W.L..next_year ~ Age.bat + PA + AB + H.bat + X2B + X3B +
                # HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                # TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                # GF + SHO + SV + IP + H.pitch + HR.pitch +
                # BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                # FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                # G + Inn + Ch + PO + A + E + DP + Fld. +
                # RF.9 + RF.G + tot_fa_war3 + num_fas || Tm, data = train.df, verbose=TRUE)
```

```r
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$BA, 2, raw = FALSE) |
##         train.df$Tm))
##
## REML criterion at convergence: 3545.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.0813 -0.6780 -0.0020  0.6808  3.1719
##
## Random effects:
##  Groups        Name                              Variance  Std.Dev.  Corr
##  train.df.Tm   (Intercept)                       1.137e+01 3.3716324
##  train.df.Tm.1 poly(train.df$BA, 2, raw = FALSE)1 1.858e-07 0.0004311
##                poly(train.df$BA, 2, raw = FALSE)2 1.262e-08 0.0001124 1.00
##  Residual                                        4.083e+01 6.3895753
## Number of obs: 536, groups:  train.df$Tm, 29
##
```

```
## Fixed effects:
##                                  Estimate Std. Error t value
## (Intercept)                       50.1940     0.6892  72.827
## poly(train.df$BA, 2, raw = FALSE)1  35.8076     8.4294   4.248
## poly(train.df$BA, 2, raw = FALSE)2   6.7245     6.7529   0.996
##
## Correlation of Fixed Effects:
##                     (Intr) p(.$BA,2,r=FALSE)1
## p(.$BA,2,r=FALSE)1 0.010
## p(.$BA,2,r=FALSE)2 0.005  0.016
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + (1 + WHIP + W.L..same_year +
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.971771
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.244459
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch | Tm, data = train.df)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.872679
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.224486
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + tot_fa_war3 | Tm, data = tra
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.750696
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.259757
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + H9 | Tm, data = train.df)
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.644522
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.153371
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + H9 + (1 + WHIP + W.L..same_y
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.714458
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.13684
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ Age.bat + PA + AB | Tm, data = train.df)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 2 negative eigenvalues
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.903281
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.389564
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ Age.bat + PA + AB + (1 + Age.bat + PA + AB | Tm) , data = train.
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 4 negative eigenvalues
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.953489
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.31225
```